Flexible Service Provision in Context-Aware Cyber-Physical Systems

Ramón Alcarria[1,2(✉)], Borja Bordel[1,2], and Antonio Jara[1]

[1] Institute of Information Systems, University of Applied Sciences Western Switzerland, Delémont, Switzerland
ramon.alcarria@upm.es, bbordel@dit.upm.es, jara@ieee.org
[2] Universidad Politécnica de Madrid, Madrid, Spain

Abstract. Cyber-Physical Systems have emerged in recent years as a new technological revolution to support a collection of devices in the execution of physical processes. In mobility scenarios the execution context is in continuous change. Thus, Context-aware Cyber-Physical Systems must be adapted to this situation to provide services optimally. This paper proposes mechanisms for profile selection and consistency checking for a flexible service provision in context-aware cyber-physical systems. These mechanisms are validated through an experimentation, by studying the reaction of the proposed system to changes in context.

Keywords: Cyber-Physical systems · Service provision · Matchmaking · Service resolution

1 Introduction

The term Cyber-Physical Systems (CPS) refers to the connection of embedded devices as integrations of computation and physical processes [1]. In order to provide a service-oriented architecture for CPS the so-called service-oriented CPS emerged as physical and logic infrastructures offering a collection of services and capabilities to enable their publication, discovery and execution.

In scenarios involving changes in the number and nature of devices, as well as in the environment around them, mechanisms are needed for the recognition and connection of new devices [2] and for a seamless service provision. The contribution of this paper is described in the context of context-aware CPS, proposing improvements in the resolution process, that is, the process by which certain functionalities provided by a low-level device can be substituted by others provided by that device or other, so that service execution is preserved in a seamless way.

The resolution mechanisms proposed in this work are, firstly, a detector and applicator of execution profiles for the services provided by cyber physical devices,

R. Alcarria and B. Bordel—On leave from Universidad Politécnica de Madrid, Madrid, España.

L. Barolli and T. Enokido (eds.), *Innovative Mobile and Internet Services in Ubiquitous Computing*, Advances in Intelligent Systems and Computing 612,
DOI 10.1007/978-3-319-61542-4_89

each of these profiles activates a set of functionalities that are offered to the service orchestrator at a higher abstraction level. Secondly, a consistency checker between running services and the execution environment.

The rest of the paper is as follows. Section 2 describes some related work in context-aware CPS and Service Provision. Section 3 shows some background on Context-aware CPS. Section 4 provides the main contribution of our work in service profile detection and enablement, and consistency checking. Finally, Sects. 5 and 6 show an experimental validation and some conclusions.

2 Related Work

This section describes some works related to context-aware CPS, and how these and other works deal with service provision and resolution processes.

Related to context-aware CPS, some approaches have integrated traditional CPS networks, such as vehicular networks, with the context-aware concept [3]. CPS interacting with real life's surroundings have specific needs when using cloud-based solutions. These needs are studied in [4]. Other works investigate the extent to which context information may be used to improve security and survivability of CPS [5].

Service provision has also been studied in the field of CPS. The work of Mikusz [6] explores the utility of CPS for the provision of value to customers, through the concept of Service-Dominant Logic. Other works deal with the service delegation process [7] considering the concept of task, a collection of actions to be executed in cloud environments or in underlying devices. Finally, technologies for the interconnection of components such as OSGi allow a flexible composition of services, taking into account the possible connections and disconnections of the devices that provide them [8].

The resolution process is defined in the literature as the process by which the description of a service (abstract service) is associated with the service provided by a device (specific service) [8]. The service resolution process must decide which implementation is most appropriate in a given context, according to the access technology [9], if inconsistencies occur [10], or if the execution profile of the device is adequate.

Finally, in relation to CPS and the convergence to Internet of Things, some complex issues such as self-configuration of heterogeneous hardware platforms or services composition, access and service managements [11] are currently being much studied. Many solutions related to service composition are based on Business Process Execution Language (BPEL) workflows [12], where executing tasks calls some external services, often offered by devices. Service access in CPS is commonly tacked by the provision of an access middleware [9]. Two solutions are proposed: embedded TCP/IP stacks in all devices [13] and translation layers adapting web services provided by applications to the particular interfaces of devices [14].

3 Context-Aware Cyber-Physical Systems

The proposed architecture for contributing to resolution process in context-aware CPS is presented in Fig. 1.

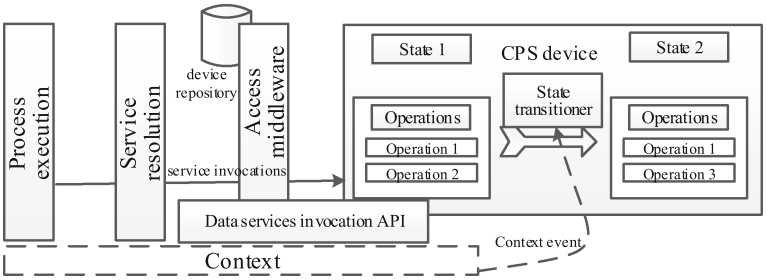

Fig. 1. Architecture for process execution in context-aware CPS

The process execution module presents an execution engine that follows a workflow-based logic for the execution of a cyber-physical process. This process generates invocations to functionalities offered by devices. To determine the optimal device to execute the functionality requested by the process, the resolution process is performed. When the resolution process determines the optimal device invocation occurs through the access middleware by the data services invocation API.

We model a CPS device with the following abstractions:

- State, representing a unique configuration of information in device. It integrates a complete set of properties (name-value pairs) that represents something relevant and significant for the other services to know and operations, i.e. the functionalities provided by this device under a given state.
- Events: The execution context is modeled as an event generator that describes a situation. Events are initiated either by user actions on the UI, by requests from other services or by results from invocations to IoT objects. The state transitioner gets these events and produce state changes in the CPS device.
- Operations: Represent functionalities associated to device states. Functionalities are often invocations to IoT elements and local capabilities (functionalities provided by executing nodes or devices) and also to the provision of graphical elements such as user interfaces. Usually, operations also cause state changes and the generation of the corresponding context events.

4 Enabling Flexible Service Provision

To contribute to a better service provision, so that context changes are contemplated and supported in the execution plan, we plan to contribute in the areas of profile-based execution and consistency checking.

To ensure a seamless service execution we propose the invocation of functionalities depending on the state of the devices. For this we propose the concept of execution profiles. Each profile activates a set of functionalities that are offered to the orchestrator. The resolution process supports changes in the profiles, produced by changes in the context in which these cyber-physical devices operate.

To ensure the compatibility of the services in execution we developed a consistency checker, which detects the problematic service and shuts it down or substitutes it for a similar one composed by rules consistent with the current situation.

4.1 Profile Detection and Execution

The first mechanism is a detector and applicator of execution profiles for the services provided by cyber-physical devices. Cyber-physical devices may have been programmed to operate under different execution profiles. Each profile activates a set of functionalities that are offered to the orchestrator. The matchmaking mechanism has to support changes in the profiles, produced by changes in the context in which these cyber-physical devices operate.

We consider CPS devices as transducers, converting variations in a physical quantity, such as pressure or brightness, into an electrical signal, or vice versa, and their hardware controllers, for management and processing capabilities [2].

The plug-and-play architecture described in Fig. 2(a) can be implemented in any controller or transducer, with the proviso that programming capability and a Serial Peripheral Interface (SPI) (the most common in microelectronics) must be available. SPI ports allow full-duplex communications using four independent lines:

- Master Output Slave Input (MOSI) line, to transmit data from the master device to the slave device.
- Master Input Slave Output (MISO) line, to transmit data from the slave device to the master device.
- Clock (CLK) line, for sharing the synchronization signal.
- Slave Select (SS) line, to active the slave device.

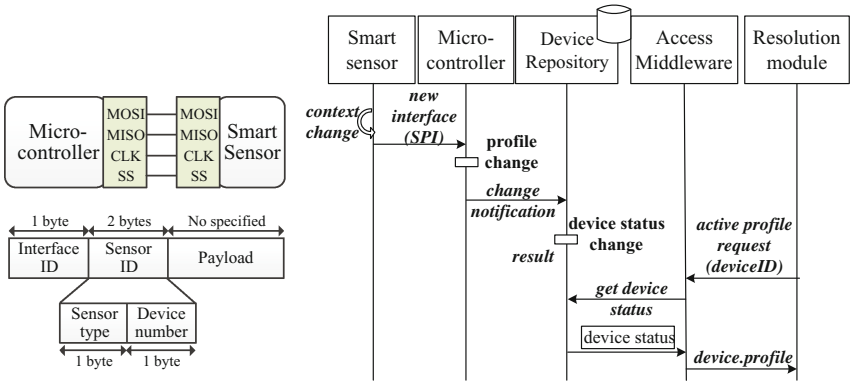

Fig. 2. (a) Microcontroller-sensor communication, (b) interaction diagram

For the communication between these two elements we consider the standards family IEEE 1451. This specification set addresses the problem of supporting a plug-and-play architecture based on transducers and microcontrollers [15], providing a

common interface for connecting transducers and processors to controllers and data acquisition systems. For profile discovery and selection we employ the standardized Transducer Electronic Data Sheets (TEDS) specified by IEEE 1451.2, which allows the self-description of interfaces, and the interaction to hardware controllers.

The microcontroller detects the context, either from the sensors or from the cloud, and proceeds to change the profile of the sensors it has connected.

When the context change occurs from the sensors, in the SPI interfaces, the master device must activate the Slave Select signal (SS) during a time equal to the time required by the slave device to perform the transmission of its new data. New data coming from the smart sensors is transmitted through the MISO port. Information transmitted is: *InterfaceID*, *SensorID* and *Payload*.

As described in Fig. 2(b), once received this information, the microcontroller proceeds to change the current profile of the connected sensor. Information about CPS device is store in the device repository database. Once the resolution requests the active profile of a given sensors, it ask the device repository through the access middleware.

An equivalent interaction is performed when the microcontroller detects the context from others systems or cloud environments, and must notify the sensor for a change of interface. In this case the MOSI channel is used and a message with a structure similar to the previous one is sent.

4.2 Consistency Checker

In rule-based cyber-physical environments, there may be inconsistencies between rules, which are resolved by disabling problematic services. We design our consistency checker based on LTL rules, describing the functions that can be performed by CPS devices and the constraints prohibiting undesired behavior.

LTL formulas are composed of a finite set of atomic propositions, the Boolean connectives $\neg$ (not), $\wedge$ (and), $\vee$ (or), and $\rightarrow$ (implies), and the temporal connectives $\mathfrak{U}$ (until), $\Re$ (release), $\boxdot$ (next time), $\sqcup\!\!\sqcup$ (globally) and $\diamondsuit$ (in the future). Intuitively, $\phi \,\mathfrak{U}\, \psi$ states that ϕ remains true until ψ holds. Dually, $\phi \,\Re\, \psi$, or ϕ releases ψ, means that ψ must be true now and remain true until ϕ is true, thus releasing ψ. $\boxdot\, \phi$ means that ϕ is true in the next time step after the current one. Finally, $\sqcup\!\!\sqcup\, \phi$ commits ϕ to being true in every time step while $\diamondsuit\, \phi$ designates that ϕ must either be true now or at some future time step.

The consistency checker converts the LTL variability restrictions to finite state automata and analyses the constraint model generated in real-time, determining if the construction of this model by user actions results in a change from *satisfied* to *temporarily violated* and *permanently violated*.

The consistency checker is also used to resolve circular dependencies. Suppose that the resolution process selects a functionality offered by a device A which in turn requires a device B to operate. Suppose the running service makes a call to a C functionality, which requires B to be turned off for proper service execution. By modelling these constraints through LTL formulas, this circular dependency is automatically recognized and extracted.

This work models the ability of the devices to execute services according to the quality of service they offer for each of the profiles they have implemented. Information about QoS is stored in the device repository, and is presented as a k number of n-dimensional vectors of integer values, indicating the value of a collection of quality parameters (availability, response time, etc.), being k the total number of profiles implemented in the device.

$$QoS_{offer_k} = (\alpha_{1_k}, \alpha_{2_k}, \ldots, \alpha_{n_k}) \tag{1}$$

On the other hand, the process execution model handles a collection of quality parameters for each task, as follows:

$$QoS_{request} = (\beta_1, \beta_2, \ldots, \beta_m) \tag{2}$$

As in the previous case, for each task the indicated quality parameters may not be the same, and the dimension of the vector $QoS_{request}$ can also vary.

The resolution process uses Algorithm 1 to combine the constraints imposed by the quality of service with those imposed by the consistency detector.

Algorithm 1 Resolution process

Output: Resolved process *resprocessId*and *processStatus*
Define List<*deviceC*> as validC

for each*serviceId* from process *processId*
validC = getValidC (*serviceId, $QoS_{request}$*)
if (validC is not empty)
if *(active profile QoS belongs to* validC*)*
processStatus = Status.Available
resolveService (serviceId, validC)
else
processStatus = Status.TempUnavailable
end if
else*processStatus* = Status.Unavailable
end if
end for

Function getValidCandidates
Input: Service*serviceId* to resolve
Output: List<*deviceC*> List of device candidates ordered by QoS

Define List<*deviceC*> as initialCandidates
Define List<*deviceC*> as consistentC
Define List<*deviceC*> as consistent&QoSC
Define List<QoS_{offer}> as QoSofferList

initialCandidates= getC (*$QoS_{request}$*)
for each icandidate from *initialCandidates*
if (checkConsistency (icandidate, C_ϕ)
then store icandidate in *consistentC*
end if
end for

for each ccandidate from *consistentC*
QoSofferList = getAllProfileOffers(ccandidate);
for eachQoS_{offer} in QoSofferList
if (QoS_{offer}>*$QoS_{request}$*)
then store ccandidate in *consistent&QoSC*
end if
end for
sort List *consistent&QoSC* by QoS_{offer}
return*consistent&QoSC*

The operation of the algorithm is indicated below. For all the services composing a process, the resolution process is executed, which searches for candidate devices to execute the functionality defined by the process. These candidates must comply with the consistency requirement, i.e., that the consistency checker validates that there is no inconsistency with the rest of the services in execution and with the current context, and the quality of service requirement, whereby the quality offered by the device must

be greater than the quality required by the process ($QoS_{offer} > QoS_{request}$). As the quality offered by the device depends on the execution profile that the device has at the moment, a comparison must be made for all the profiles, so that the execution engine is always aware of the current state of the resolution process and of the possibility of execution of certain services in compatible devices, if there is a change of context that favors an increase of the quality of service in the candidate device. If the active profile of the device has sufficient QoS (according to the interaction diagram in Fig. 2(b)) the service can be executed on the device.

According to the result of the resolution process, we associate to the process a process state. Thus, each process may present three different states: available, temporarily unavailable and unavailable. Table 1 describes the conditions which must fit a process to present each state.

Table 1. Process' states in Context-aware CPS.

State	Description
Available	A process is available when all the services which compose it are available, and the guaranteed QoS of the composite service surpasses the established QoS thresholds
Temporarily unavailable	A process is temporarily unavailable when at least one of the services which compose it is temporal unavailable, because the guaranteed QoS of the process falls below the established QoS thresholds in the given context
	A process which is considered temporarily unavailable may return to the available state without refreshing the composition process
Unavailable	A process is considered as unavailable when at least one service is unavailable or temporarily unavailable for a time over a certain limit
	A process which is considered unavailable must refresh the resolution process in order to change its state

5 Experimental Validation

These mechanisms are implemented on a cyber-physical environment, constructing a prototype consisting of a controller connected to two smart sensors, as can be seen in Fig. 3.

We have used a Zigbee transmitter and receiver module for the communication of two Smart sensors (*controller #1* and *controller #2*) through SPI interface. For the control of both sensors a multiplexer is used that intervenes in the slave select line, following the standard IEEE 1451. The *Smart Sensor #1* consists of a controller (*controller #1*), an NFC Reader, an NFC Antenna, potentiometer that acts as a pull-down resistor for level control and a relay, which controls the power supplied to the NFC antenna, so that the controller can cut the power consumed in periods of inactivity.

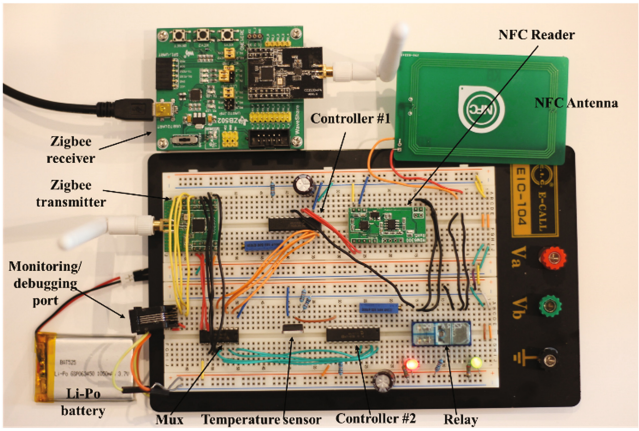

Fig. 3. Implemented prototype

The Smart Sensor #2 consists of another controller (controller #2), a temperature sensor and another potentiometer for regulating the reference levels of the A-D converters that control the temperature level.

The prototype also has a rj-11 port for monitoring and debugging.

As the consistency checker module have medium processing requirements we opted to implement it in the cloud environment (specifically by a JSP application installed in a t2.mini Amazon EC2 instance), reducing the complexity of the resolution process, which is implemented in the microcontroller based-on TI CC2530.

The prototype was deployed in a laboratory of the Technical University of Madrid, where various emulated production systems were implemented. We provide some logic to the *authentication* cyber-physical device managed by Controller #1. It has two profiles *fullOperation* and *deactivated*. In the *fullOperation* profile the NFC antenna is always switched on, and in the *deactivated* profile, the relay permanently deactivates the antenna, so authentication is disabled. These two profiles implements the same openDoor interface, in the fullOperation profile the openDoor function includes an authentication process by using an NFC tag. In the deactivated profile, the *openDoor* function returns always true, without any authentication process.

We also configure Controller #2 to generate context events so that CPS devices can change their state such as *temp.LOW*, *temp.MEDIUM*, *temp.HIGH*.

We perform an experiment to validate the contributions of our paper, specifically, profile detection and execution and consistency checking in the resolution process. To do that we include the following LTL rule in the consistency checking module:

$\sqcup\!\!\sqcup$ (*temp.HIGH* $\rightarrow$ $\Diamond$ *NFC.deactivated*) An event informing of a high temperature recording must be followed in the future by a deactivation of the NFC reader.

We simulate an increase in the temperature to generate*temp.HIGH* and after that we measure the time it takes for the resolution process to select the *authentication* cyber-physical device as the optimal device to perform the *openDoor* function.

We divide the resolution process in steps and measure the time it takes for the completion of each step:

1. Selection of initial candidates: The resolution module connects to the device repository through the access middleware and retrieves the information about devices, including QoS_{offer_k} parameters.
2. Checking candidate consistency: The consistency checker compares the functions offered by the devices with the consistency rules of the CPS, detecting inconsistencies and circular dependencies.
3. GetProfiles from candidates: $QoS_{request}$ is compared with all QoS_{offer} so that only a set of profiles is selected.
4. Checking of active profile: We check which of the candidates has an active profile that meets the conditions of execution
5. Set process status: depending on the above we classify the process as available, temporarily unavailable or unavailable.

We execute many resolution processes (1, 5 and 10) in parallel to simulate more components in the cyber-physical system. Results can be seen in Table 2.

Table 2. Average execution time in *ms* of steps #1 to #5.

Step	Average execution time of Steps #1 to #5		
	1 process	5 processes	10 processes
#1	453 ms	502 ms	633 ms
#2	501 ms	1107 ms	2165 ms
#3	113 ms	140 ms	201 ms
#4	39 ms	41 ms	51 ms
#5	15 ms	17 ms	19 ms
Total	1121 ms	1807 ms	3069 ms

As can be seen, the execution of the resolution process for a component while maintaining the system with a load of 10 processes in parallel has an average duration of 3069 ms, with a standard deviation of 114 ms. This is the maximum response time of the system, which is conditioned by the hardware limitations of the prototype, such as the processing capacity of the TI CC2530 chip, which manages most of the resolution process, except the consistency checking, which is performed in the Cloud infrastructure.

With the results of this experiment it can be seen that step #2 is the one that increases most in duration as parallel processes are added. The explanation is that we consider the connection time to the cloud infrastructure through the Zigbee channel and a PC that controls the IP communication.

6 Conclusions

In this paper we have presented the problem of process resolution in Cyber-Physical Systems and we have devised a resolution mechanism that allows to select the most appropriate cyber-physical device to the current situation for a flexible execution in changing environments. The resolution process integrates two main contributions. The first is the detection and application of execution profiles, defined as different modes of behavior that intelligent devices have, which depend on the context in which they are executed.

The second contribution focuses on the detection of inconsistencies and circular dependencies through a declarative logic based on LTL, to prevent malfunctions in a continuous process execution.

We describe the resolution process through a functional architecture and an algorithm, and we have implemented the functional modules in a hardware prototype, which we have validated in a use case. The results on the measurements show execution times of the resolution process of little more than 1.1 s when there is a single process running and of little more than 3 s for the existence of 10 processes, in both cases for the hardware selected for the tests. With these results we can say that the service provision process proposed in this paper is sufficiently flexible for Context-aware CPS.

Acknowledgments. The research leading to these results has received funding from the Ministry of Economy and Competitiveness through INPAINK (RTC-2016-4881-7) and SEMOLA (TEC2015-68284-R) projects. Borja Bordel has received funding from the Ministry of Education through the FPU program (grant number FPU15/03977)

References

1. Lee, A.: Cyber-physical systems—are computing foundations adequate? In: NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap, Austin, TX, USA, 16–17 October 2006
2. Bordel, B., Sanchez-de-Rivera, D., Alcarria, R.: Plug-and-play transducers in cyber-physical systems for device-driven applications. In: 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 316–321 (2016)
3. Wan, J., Zhang, D., Zhao, S., Yang, L.T., Lloret, J.: Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions. IEEE Commun. Mag. **52** (8), 106–113 (2014)
4. Berger, C.: Cloud-based testing for context-aware cyber-physical systems. Softw. Test. Cloud Pers. Emer. Discipline, 68–95. IGI Global (2013). Accessed 4 Jan 2017
5. Wan, K., Alagar, V.: Context-aware security solutions for cyber physical systems. Mob. Netw. Appl. **19**(2), 212 (2014)
6. Mikusz, M.: Cyber-physical systems as service systems: implications for S-D logic. In: The 2015 Naples Forum on Service, pp. 1–19 (2015)
7. Sztipanovits, J., Koutsoukos, X., Karsai, G., Kottenstette, N., Antsaklis, P., Gupta, V., Wang, S.: Toward a science of cyber–physical system integration. Proc. IEEE **100**(1), 29–44 (2012)

8. Alcarria, R., Robles, T., Domínguez, A.M., González-Miranda, S.: Flexible service composition based on bundle communication in OSGi. KSII Trans. Internet Inf. Syst. **6**(1), 116–130 (2012)
9. Alcarria, R., Valladares, T.R., Morales, A., Ipiña, D.L., Aguilera, U.: Enabling flexible and continuous capability invocation in mobile prosumer environments. Sensors **12**(7), 8930–8954 (2012)
10. Robles, T., Alcarria, R., Morales, A., Martín, D.: Supporting variability dependencies for rule-based service compositions in prosumer environments. Int. J. Web Grid Serv. **11**(1), 57–77 (2015)
11. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
12. Spiess, P., Karnouskos, S., Guinard, D., Savio, S., Baecker, O., Souza, L., Trifa, V.: SOA-based integration of the internet of things in enterprise services. In: Proceedings of IEEE ICWS, Los Angeles, USA, July 2009
13. Duquennoy, S., Grimaud, G., Vandewalle, J.J.: The web of things: interconnecting devices with high usability and performance. In: Proceedings of ICESS 2009, HangZhou, Zhejiang, China, May 2009
14. Buckl, C., Sommer, S., Scholz, A., Knoll, A., Kemper, A., Heuer, J., Schmitt, A.: Services to the field: an approach for resource constrained sensor/actor networks. In: Proceedings of WAINA 2009, Bradford, United Kingdom, May 2009
15. Song, E.Y., Lee, K.: Understanding IEEE 1451-networked smart transducer interface standard-what is a smart transducer? IEEE Instrum. Measur. Mag. **11**(2), 11–17 (2008)